Project 2:

Submission deadlines: 5:00 pm, Friday 21st May 2021
<u>Value:</u> **15%** of CITS1401.

To be done individually.

You should construct a Python 3 program containing your solution to the following problem and submit your program electronically on Moodle. No other method of submission is allowed. Your program will be automatically run on Moodle for sample test cases provided in the project sheet if you click the "check" link. However, your submission will be tested thoroughly for grading purpose by teaching team after the due date. Remember you need to submit the program as a single python file and copy-paste the same program in the provided text box. You have only one attempt to make the submission so don't submit if you are not satisfied with your attempt. All open submissions at the time of deadline will be automatically submitted. There is no way in the system to open the closed submission and reverse your submission.

You are expected to have read and understood the University's <u>guidelines on academic conduct</u>. In accordance with this policy, you may discuss with other students the general principles required to understand this project, but the work you submit must be the result of your own effort. Plagiarism detection, and other systems for detecting potential malpractice, will therefore be used. Besides, if what you submit is not your own work then you will have learnt little and will therefore, likely, fail the final exam.

You must submit your project before the submission deadline listed above. Following UWA policy, a late penalty of 5% will be deducted for each day (24 hours), after the deadline, that the assignment is submitted. No submissions will be allowed after 7 days following the deadline except approved special consideration cases.

Overview

This project is an extension to Project 1 but has different requirements. As mentioned earlier that the year 2020 will be regarded as a pandemic year in the history of mankind. COVID-19 impacted the entire world in such a manner that no other virus has ever done in the history. It's been more than a year for the virus and still the uncertainties are looming all over the world.

Center for Systems Science and Engineering at John Hopkins University is regularly gathering the data about the COVID-19 spread and publishing it regularly at https://ourworldindata.org/coronavirus-source-data. This data is sourced from governments, national and subnational agencies across the world and publicly available for researchers and analysts.

In this project, you are required to write a computer program which can read the data from a csv (comma separated values) file provided to you and return different analytical results of the COVID-19 cases for entire world. Your program should follow the following specifications.

CITS1401 Computational Thinking with Python
Project 2 Semester 1 2021

Specification: What your program will need to do

Input:

Your program must define the function **main** with the following signature:

```
def main(csvfile):
```

The input arguments are:

- `csvfile` is the name of the CSV file containing information and record of the COVID-19 cases which needs to be analysed. The first row of the CSV file will contain the headers. From the second row, each row contains the corresponding data to its header. We do not have prior knowledge about the number of columns or rows available in the CSV file.

Output:

The function is required to return the following outputs in the order provided below:

- A dictionary containing the country name as *key* and a list having the following data about that country as *value*.
  - A list containing the total number of recorded positive cases of COVID-19 for each month of the year.
  - A list containing the total number of recorded deaths due to COVID-19 for each month of the year.
  - A list containing the total number of days for each month of year, when the recorded positive cases of COVID-19 for that month of the year were greater than the average recorded positive cases of that month of the year.
  - A list containing the total number of days for each month of year, when the recorded deaths due to COVID-19 for that month of the year were greater than the average deaths due to COVID-19 for that month of the year.
- A dictionary containing the continent name as *key* and a list having the data similar to above for each continent as *value*.

All lists should have values recorded for each month of the year in order from January to December. All returned output variables must contain numerical values rounded to four decimal places (if required to be rounded off). Remember not to round the values during calculations and round them only at the time of saving them in the output variables.

Example:

Download the `Covid-data-for-project_2_sample.csv` file from the folder of Project 2 on LMS or Moodle. An example interactions are:

```
>>> dict_country,dict_continent = main('Covid-data-for-project-2-sample.csv')
```

The output returned are dictionaries where some data is presented below:

```
>>> dict_country['afghanistan']
```

```
[[1963, 1, 174, 1952, 13081, 16020, 1681, 1494, 1109, 2157, 4849, 5252], [86,
0, 4, 60, 194, 482, 188, 119, 57, 78, 257, 396], [1, 1, 7, 11, 13, 16, 8, 11,
10, 14, 15, 18], [3, 0, 4, 12, 11, 13, 8, 15, 14, 15, 15, 16]]
```

```
>>> dict_country['italy']

[[169327, 1126, 104664, 99671, 27534, 7729, 6959, 21677, 45647, 364569, 922124,
505612], [4596, 29, 12399, 15539, 5448, 1383, 374, 342, 411, 2724, 16958,
18583], [8, 8, 17, 15, 11, 15, 14, 12, 15, 11, 15, 15], [6, 6, 14, 17, 12, 16,
13, 4, 15, 12, 15, 15]]

>>> dict_continent.keys()

dict_keys(['asia', 'europe', 'africa', 'north america', 'south america',
'oceania', ''])

>>> dict_continent['oceania']

[[198, 16, 4534, 2207, 436, 718, 9360, 8539, 1277, 499, 317, 513], [0, 0, 18,
75, 10, 2, 97, 456, 231, 19, 1, 1], [10, 5, 10, 11, 13, 9, 14, 13, 13, 13, 10,
12], [0, 0, 12, 14, 10, 2, 11, 14, 10, 10, 1, 1]]
```

Additional requirements:

There are few more requirements for your program.

- Your program needs to validate the inputs to the main() function and gracefully terminate if invalid inputs are provided.
- You program needs to terminate *gracefully* if the file cannot be found or opened.
- For graceful terminations, you need to print the message related to the problem and return None for each output.
- Your program needs to validate the input data from the file. Date in the file is expected to be in day/month/year format otherwise entire row needs to be discarded. All numeric data is expected to be numeric and any other recorded data (or no data) should be considered as zero (0).
- Your program needs to consider that columns and rows of the csv file do not have any specific order or can be in any order (excluding header row).
- Your program needs to interpret the header row to find the required columns. Your program needs to terminate *gracefully* if required columns cannot be found in the file.
- Your program needs to convert all text data in the csv to be lower order alphabets.
- Your program needs to consider that data can be missed for few days of the month and therefore needs to find averages based on the available number of days for which correct data is available (for output related to countries), whereas for continents, the program can assume that data exists for all days of the month.

Important grading instruction:

You will have noticed that you have not been asked to write specific functions. That has been left to you. However, it is important that your program must defines the top-level function main(). The idea is that within main(), the program calls the other functions. (Of course, these may call further functions.) The reason this is important is that when your program is tested, the testing program will call your main() function. So, if you fail to define main(), the testing program will not be able to test your program and your submission will be graded zero. Don't forget the submission guidelines provided at the start of the project sheet.

CITS1401 Computational Thinking with Python
Project 2 Semester 1 2021

Things to avoid:

There are a few things for your program to avoid.

- You are not allowed to import any Python module. While use of the many of these modules, e.g. csv or math is a perfectly sensible thing to do in a production setting, it takes away much of the point of different aspects of the project, which is about getting practice opening text files, processing text file data, and use of basic Python structures, in this case lists and loops.
- Do not assume that the input file names will end in .csv. File name suffixes such as .csv and .txt are not mandatory in systems other than Microsoft Windows.
- Ensure your program does NOT call the `input()` or `print()` functions at any time (`print()` function can be used for graceful terminations only). That will cause your program to hang, waiting for input that automated testing system will not provide. In fact, what will happen is that the marking program detects the call(s), and will not test your code at all which may result in zero grade.

Submission:

Submit your solution before the deadline electronically on Moodle. No other method of submission is allowed. Your program will be automatically run on Moodle for sample test cases provided in the project sheet if you click the "check" link. However, your submission will be tested thoroughly for grading purpose by teaching team after the due date. Remember you need to submit the program as a single Python file and copy-paste the same program in the provided text box. You have only one attempt to make the submission so don't submit if you are not satisfied with your attempt. You are encouraged to keep your attempt open as there is no way in the system to open the closed submission and reverse your submission. All open submissions at the time of deadline will be automatically submitted except special consideration or late submissions.

You need to contact unit coordinator if you have special considerations or making submission after the mentioned due date.

Marking Rubric:

Your program will be marked out of 30 (later scaled to be out of 15% of the final mark).

22 out of 30 marks will be awarded based on how well your program completes a number of tests, reflecting normal use of the program, and also how the program handles various states including error states, different number of rows in the input file or missing data for months/days, etc. You need to think creatively what your program may face. Your submission will be graded by data files other than the provided sample data file. Therefore you need to be creative to look into corner or worst cases. I have provided few guidelines from ACS Accreditation manual at the end of the project sheet which will help you to understand the expectations.

8 out of 30 marks will be awarded on *style* (5/8) "the code is clear to read" and *efficiency* (3/8) "your program is well constructed and runs efficiently". Remember, the expected standards are much higher than those of earlier project.

For style, think about use of comments, sensible variable names, your name and student ID at the top of the program, etc. (Please watch lectures where this is discussed.)

CITS1401 Computational Thinking with Python
Project 2 Semester 1 2021

Style Rubric:

0	Gibberish, impossible to understand
1-2	Style is really poor or fair
3-4	Style is good or very good, with small lapses
5	Excellent style, really easy to read and follow

Your program will be traversing text files of various sizes (possibly including large csv files) so you need to minimise the number of times your program looks at the same data items.

Efficiency Rubric:

0	Code too incomplete to judge efficiency, or wrong problem tackled
1	Very poor efficiency, additional loops, inappropriate data reading or use of `readline()`
2	Acceptable or good efficiency with some lapses
3	Excellent efficiency, should have no problem on large files, etc.

Automated testing is being used so that all submitted programs are being tested the same way. Sometimes it happens that there is one mistake in the program that means that no tests are passed. If the marker is able to spot the cause and fix it readily, then they are allowed to do that and your - now fixed - program will score whatever it scores from the tests, minus 4 marks, because other students will not have had the benefit of marker intervention. Still, that's way better than getting zero. On the other hand, if the bug is hard to fix, the marker needs to move on to other submissions.

Extract from Australian Computing Society Accreditation manual 2019:

As per Seoul Accord section D,

A complex computing problem will normally have some or all of the following criteria:

- involves wide-ranging or conflicting technical, computing, and other issues;
- has no obvious solution, and requires conceptual thinking and innovative analysis to formulate suitable abstract models;
- a solution requires the use of in-depth computing or domain knowledge and an analytical approach that is based on well-founded principles;
- involves infrequently-encountered issues;
- is outside problems encompassed by standards and standard practice for professional computing;
- involves diverse groups of stakeholders with widely varying needs;
- has significant consequences in a range of contexts;
- is a high-level problem possibly including many component parts or sub-problems;
- identification of a requirement or the cause of a problem is ill defined or unknown.